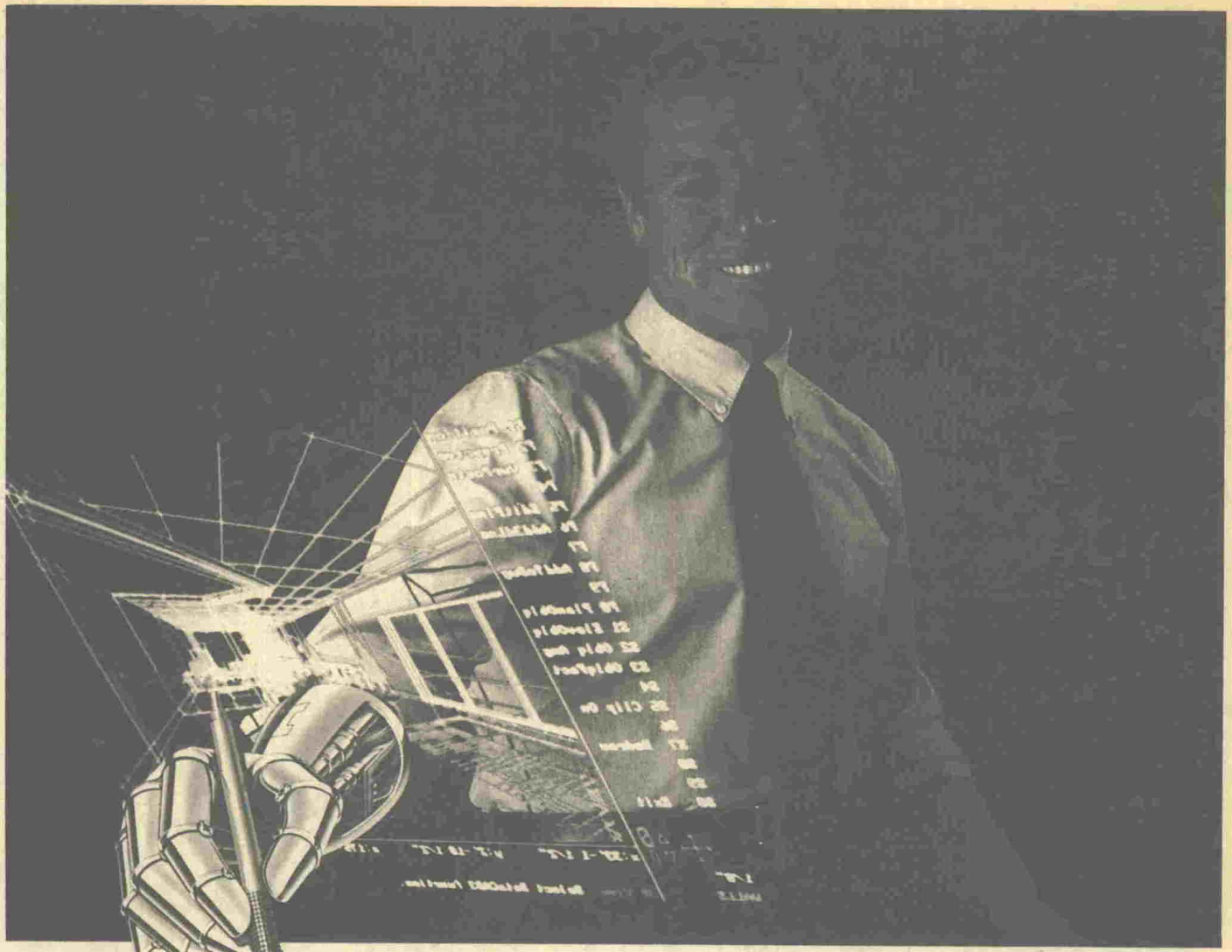


เขียน ให้ดูดี

หลักการเบื้องต้น
สำหรับเขียนโปรแกรมภาษา C
ให้ดูทะมัดทะแมง เข้าใจง่าย

ภุชงค์ อุทโยภาศ*

เร็ว ๆ นี้ ผมไปเจอบทความของฝรั่งบทหนึ่ง กล่าวถึงการเขียนโปรแกรมภาษา C อย่างถูกหลักว่าเขาเขียนกันอย่างไร ผมคิดว่าในตอนนี้ท่านผู้อ่านหลายท่านคงจะเล่นภาษา C กันอยู่ ขอแอบเก็บความมาซุบซิบเล่าสู่กันฟังก็แล้วกันนะครับ เผื่อจะเป็นประโยชน์บ้าง

เขาบอกว่า ถ้าจะเขียนโปรแกรมภาษา C ให้คนอื่นอ่านรู้เรื่องกะเราบ้างนั้น ต้องทำอย่างนี้คือ

1. ใช้เทคนิคการเขียนโปรแกรมแบบโครงสร้าง

เนื่องจากภาษา C นั้น สร้างขึ้นมาจากฟังก์ชันต่าง ๆ หลายฟังก์ชันมาต่อกันเป็นโปรแกรม ดังนั้นพยายามเขียนโปรแกรมโดยกระจายออกเป็นฟังก์ชันย่อย ๆ ซึ่งมีการเรียกใช้กันเป็นระดับ โดยแต่ละฟังก์ชันนั้น ควรจะเขียนให้

*ชมรม IEEE
สาขานิสิตจุฬาลงกรณ์มหาวิทยาลัย

กระจ่างแจ้งว่า ทำงานอย่างไร และทำอะไร อย่าพยายามเขียนฟังก์ชันซึ่งทำงานหลายอย่างพร้อมกัน หลักสำคัญคือพยายามสร้างโปรแกรมให้ประกอบไปด้วยฟังก์ชันเล็ก ๆ ซึ่งสามารถนำไปใช้งานได้ ซ้ำแล้ว ซ้ำเล่า และในแต่ละฟังก์ชันไม่ควรมีความยาวเกิน 1 หน้ากระดาษ

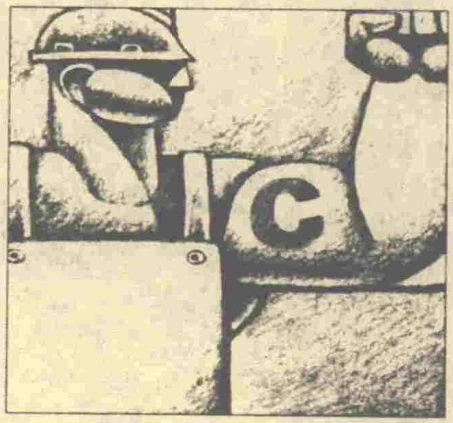
อันสุภาษิตจีนนั้น เขาว่าไว้ว่าตัวอักษรพันคำนั้น มิสู้ภาพ ๆ เดียว ดังนั้นผมเลยขอร่างรูปอธิบายข้อความตะกี้ดังรูปที่ 1

นอกจากนั้น พยายามรวบรวมฟังก์ชันในระดับเดียวกันเก็บเป็นไฟล์หรือไลบรารี่เดียวกันก็จะดีมาก

รูปที่ 1 โครงสร้างการเขียนโปรแกรมภาษา C

2. หน้าตาของซอร์สโปรแกรมภาษา C ที่ดี

หน้าตาของซอร์สโปรแกรมภาษา C ที่ดีนั้น เขามีตัวอย่างมาให้ดู ซึ่งผมก็แอบมุบมิบลอกมาลงให้ท่านผู้อ่านดูกันตรงนี้เลยครับ



■ ตัวอย่างซอร์สโปรแกรมภาษา C

```
/* prolog: this file contains functions which do the following  */
/*          copyright message, date ,programmer                 */

/* usage: you can call the program in the following way:        */
/*         program_name flags file_name_arguments .....         */

/* description: a more complete description of the things       */
/*         this program should do when called with flags and    */
/*         filenames                                            */

#include .......          /* inclusion of standard header file */
#include .......          /* inclusion of header files only used by */
                          /* this program or other local include files


/*
 *  comment all define
 */
 #define
 /*
  *  comment all declarations
  */
```

```
global  declarations
static  declarations
extern  declarations  /* can also be put in a header file when */
                      /* more than one file-module uses them   */
...........
...........

/*
 *   a comment should be used for every function
 *   telling (in short) what the function does
 *   which argument it expects
 *   which global variable is used
 *   and which return value is returned
*/
function( arg1,arg2, ... )
argument declarations     /* comment all declaration */

     local declarations /* comment all declarations */
     /*
      *   a block comment in front of some block of statements
      */
      ........... /* SOMETIMES comment on one specific statement */
     /*
      *   a block comment in front of some block of statements
      */
      ........... /* SOMETIMES comment on one specific statement */
```

โดยสรุปแล้ว ในแต่ละขั้นตอนของการทำงาน ต้องมีการใส่คอมเมนต์ที่ชัดเจนเพื่อบอกถึงชนิดของฟังก์ชัน หน้าที่ของมันการเรียกใช้งาน ตัวแปรจากภายนอกที่นำมาใช้ และค่าที่ส่งกลับเมื่อการทำงานเสร็จสิ้นลง (ถ้าแกะดูเองก็คงสรุปคล้ายผมแหละน่า)

3. การกำหนดตัวแปรและค่าคงที่

ใช้ตัวแปรในการกำหนดค่าในกรณีเหล่านี้ อย่าใช้ตัวเลขตรง ๆ เด็ดขาด

- ขนาดของอะเรย์ และบัฟเฟอร์
- สิ่งที่ขึ้นกับเครื่องคอมพิวเตอร์แต่ละรุ่น แต่ละขนาด เช่น ค่าตัวแปร int มากที่สุด (ไมโคร, มินิ, เมนเฟรม บางทีใช้ขนาดไม่เท่ากัน)
- ค่าคงที่ที่เก็บแบบตัวเลข แต่ใช้ในแง่อื่นซึ่งไม่ใช่การคำนวณ เช่น −1 คือ EOF บอกการสิ้นสุดไฟล์
- ใช้คำสั่ง Typedef ในการกำหนด

ตัวแปรที่ซับซ้อน (ตัวแปรแบบ Structure ถือว่าซับซ้อนเสมอ)

ในกรณีที่มีการนำค่าบางค่าไปใช้ในความหมายที่ไม่ใช่ตัวเลข แต่เก็บในตัวแปรแบบ char หรือ int ควรใช้คำสั่ง Typedef กำหนดชนิดขึ้นใหม่ เช่น typedef int COLOR

```
# define RED 1
# define BLUE 2
```

ให้ระวังให้มากในการกำหนดแมคโครเพราะอาจจะมีผลแทรกซ้อนที่ไม่คาดฝันเกิดขึ้น

อีกเรื่องที่ฝรั่งห้ามขาดคือ อย่าทำการแปลงรูปแบบของภาษา C โดยใช้ PREPROCESSOR เด็ดขาด เช่น การกำหนด

```
# define BEGIN {
```

ถือว่าผิดกฎสำนักอย่างร้ายแรง อาจถูกเจ้าสำนักขับออกไปเร่ร่อนในยุทธจักรได้

4. รูปแบบในการเขียนโปรแกรมควรตามแบบมาตรฐาน

ตัวอย่างเช่น ในการทำ Indent แต่ละบรรทัดในภาษา C นั้น มีแบบที่ใช้กันมากอยู่ 2 แบบ คือ

Indian Hill (เนินอินเดียนแดง)

เขาใช้กันในคัมภีร์สุดยอดภาษา C คือ *"THE C PROGRAMMING LANGUAGE"* ของ BRIAN W. KERNIGHAN และ DENNIS M. RITCHIE อันเป็นแม่บทของภาษา C ที่ใช้กัน หน้าตาของแบบนี้คล้ายๆอย่างนี้

```
if (CONDITION) {
                    ----------
                    ----------
} else {
                    ----------
                    ----------
}
```

แบบนี้เขามีสูตรว่าอย่างนี้ครับ

1. หนึ่งบรรทัดมีหนึ่งคำสั่งเท่านั้น

2. หลังวงเล็บปีกกาเปิด { ให้ขึ้นบรรทัดใหม่ โดยเยื้องไปทางขวา

3. ก่อนจบส่วนนั้น ให้ขึ้นบรรทัดใหม่และปิดด้วยวงเล็บปีกกาปิด } โดยเยื้องไปทางซ้ายมากขึ้น

แบบ Symmetric (สมมาตร)

แบบนี้หน้าตาอย่างนี้ครับ

```
if (CONDITION)
   {
   ----------
   ----------
   }
else
   {
   ----------
   ----------
   }
```

คราวนี้สูตรนี้

1. มีหนึ่งคำสั่งต่อหนึ่งบรรทัด

2. ก่อนเริ่มบล็อกให้ขึ้นบรรทัดใหม่แล้วพิมพ์เครื่องหมาย { แล้วขึ้นบรรทัดใหม่ โดย Indent ไปทางขวาให้มากขึ้น

3. ก่อนปิดด้วย } ให้ขึ้นบรรทัดใหม่

4. พอปิดบล็อกแล้ว คราวต่อไปเยื้องขวาน้อยลง

ฝรั่งเขาบอกว่าให้ใช้อย่างใดอย่างหนึ่งเป็นหลัก ห้ามคิดเองเพราะถึงเราจะคิดว่าดีกว่าของฝรั่ง แต่ชาวบ้านเขาจะไม่รู้เรื่องรู้ราวอะไรกับเราด้วย

นอกจากนั้น พยายามใส่เครื่องหมายปีกกาลงในโครงสร้างควบคุมเสมอ เช่น แบบนี้

```
for (-------) {
    my__function ( );}
```

จะมองดูดีกว่าแบบนี้

```
for (-------)
    my__function ( );
```

5. ระวังเรื่องลำดับก่อนหลังของตัวดำเนินการ (operator)

ในการเขียนโปรแกรมควรระวังจุดนี้ด้วย ถ้าไม่แน่ใจให้วงเล็บช่วย เช่น

```
i = j+k <<22;
```

อาจมีการผิดพลาด ควรใช้ `i = j+(k<<2);` จะดีกว่า ที่สำคัญคือ ระวังการใช้ = และ = = ในประโยค if ลองดูตัวอย่าง

```
if (i = = 3) {-------}
```

กับ

```
if (i = 3)   {-------}
```

ประโยคหลังนั้นผิด เพราะเป็นการย้ายค่า 3 ไปใส่ i แต่ C คอมพายเลอร์จะแปลให้โดยไม่มี error ทั้งคู่ ควรระวังให้ดีด้วย ตอนเริ่มเขียน C ผมเจอกรณีนี้หลายหนแล้ว กว่าจะหาเจอแทบตาย!

6. การกำหนดชื่อสิ่งต่าง ๆ

ในการกำหนดชื่อตัวแปร ควรให้มีความสัมพันธ์กับการใช้งาน เช่น ตัวแปรใช้นับ ให้ชื่อ COUNT แต่ต้องให้มีความแตกต่างกันในแปดตัวแรก เพราะคอมพิวเตอร์ส่วนใหญ่จะดูตรงนั้น

ชื่อฟังก์ชันในภาษา C ให้ใช้ตัวเล็ก ส่วนที่มาจากการกำหนดชนิดด้วยคำสั่ง Typedef และ # define ให้ใช้ตัวใหญ่

ตามธรรมเนียม เขาจะตั้งชื่อตัวแปรโดยใช้อักษรตัวแรกสัมพันธ์กับชนิดตัวแปรดังนี้

- *i, j, k* - ใช้กับตัวแปรเลขจำนวนเต็ม (integer) ที่เป็นดรรชนีของอะเรย์
- *p, q* - เป็นตัวชี้ (pointer)
- *n, m* - เลขจำนวนเต็ม (integer) ที่ไม่ใช้เป็นดรรชนี
- *f, x, y* - เป็น floating point (float)
- *s, t* - เป็นสตริง (String)
- *C* - เป็นชนิด Char (อักษรตัวเดียว)

มาถึงแค่นี้ คงพอควรแล้วละครับ จะเขียนมากเดี๋ยวมีกฎหมายลิขสิทธิ์แล้ว ผมจะโดนฟ้องเอา ขอให้คุณผู้อ่านทุกท่านสนุกกับการเขียนโปรแกรมภาษา C ทุกคนนะครับ สวัสดีครับ